AMENDMENTS TO THE CLAIMS

1. (currently amended) A multistage microprocessor pipeline structure for executing processing instructions comprising:

an instruction cache, an instruction buffer, a decoder, a register file, an arithmetic logic unit, and a cache memory, wherein to start execution of an instruction, the instruction is fetched from the instruction cache and loaded into the instruction buffer, and the instruction is then decoded for the operation by the decoder, and corresponding register values for execution of the instruction are read from the register file and are input to the arithmetic logic unit which executes the instruction;

width determination logic receives outputs from the decoder and determines a minimum effective processing operation width for executing each processing instruction and propagates width control data along with data for execution of the instruction through the pipeline structure for executing the processing instruction;

the microprocessor pipeline structure comprises a plurality of reduced bit width slices for execution of the instruction, wherein each slice comprises a reduced bit width portion of the register file, a reduced bit width portion of the arithmetic logic unit, and a reduced bit width portion of the cache memory, with a data carry operation proceeding from a lesser significant slice to a more significant slice, and the slices all operate in parallel when a full bit width processing operation is executed, or only a minimum required number[[s]] of slices is enabled if the width of the processing operation is determined to be narrower than a full bit width processing operation, and different slices are enabled and process data on a cycle-by-cycle basis.

2. (original) The multistage microprocessor pipeline structure of claim 1, wherein the width determination logic uses data about the length of operands stored in a register file tags module that stores value bit information about each operand in the register file, including the sign and width of each operand, and one or more leading bits of one or more bytes of each operand, which are examined for data overflow from a lesser significant slice to a more significant slice.

3. (original) The multistage microprocessor pipeline structure of claim 2, wherein the value bit information includes a sign of an operand in one bit, a register data width in bytes of the operand value in two bits, and one or more leading bits of one or more of the most significant bytes of the operand.

4. (original) The multistage microprocessor pipeline structure of claim 2, wherein the output of the decoder indicates two source registers and one destination register, and an instruction operation code, and the register file tags module and the instruction operation code are used to determine the number of slices required for executing the corresponding processing instruction, and those number of slices are enabled in subsequent cycles in the pipeline structure.

5. (original) The multistage microprocessor pipeline structure of claim 1, wherein a cache tag file stores addresses of the cache memory to write to and read from, and a width tag file stores the width of data stored in each memory address.

6. (original) The multistage microprocessor pipeline structure of claim 1, wherein the width determination logic outputs the width control bits to enable data flow and computation in the slices.

7. (original) The multistage microprocessor pipeline structure of claim 1, wherein enabling and disabling of each slice is accomplished by clock gating, where during enabling, clock signals allow data to proceed into and through a slice, and during disabling, clock signals block the flow of data into and through a slice.

8. (original) The multistage microprocessor pipeline structure of claim 2, wherein the width determination logic determines the likelihood of a data overflow being generated from a narrow slice operation by examining one or more leading bits of the operands which are stored in the register file tags module and generates one of three determinations:

no data overflow is guaranteed, and the effective operation width is determined by the width of the narrow operands;

data overflow is guaranteed, and the effective operation width must be one byte larger than the width of the narrow operands;
data overflow is possible but not certain, wherein a carry into the bits examined is propagated as a carry out.

9. (original) The multistage microprocessor pipeline structure of claim 1, wherein following execution and completion of a processing operation by the arithmetic logic unit, the width of the value of the processing operation result is determined, after which the width determination logic determines value bit information for the processing operation result by combining its sign bit, value width and one or more leading bits for its one or more leading bytes, which is then written to a destination register in the register file.

10. (currently amended) A method for reducing logic activity in the execution of an operation in a processor comprising the steps of:

selecting at least one operand associated with said operation,

looking up a width and a value of selected bits of said at least one operand,

determining a prediction of arithmetic overflow, based upon the width and the value of said selected bits of said at least one operand,

determining an effective width of said operation based upon the width of said at least one operand, a function specified by said operation, and said prediction of arithmetic overflow,

enabling the width of the resources in said processor corresponding to said effective width of said operation for executing said operation,

executing said operation over the enabled width of the resources, determining the width of the result of said operation based upon the step of executing.

11. (original) The method of claim 10, including saving the width of the result of said operation, and saving said result of said operation.

12. (original) The method of claim 10, wherein said step of looking up a width and a value of selected bits includes dedicated hardware for holding and retrieving said width and said value of selected bits.

13. (original) The method of claim 10, wherein the processor includes a register file, an arithmetic unit, a memory path, and a cache memory, and the register file, the arithmetic unit, and the cache memory are divided into a plurality of slices, each of which is of a reduced bit granularity, and the bits in all of the slice form a full width word in the processor.

14. (original) The method of claim 13, wherein at least one slice is of 8 bit granularity.

15. (original) The method of claim 13, wherein at least one slice is of 16 bit granularity.

16. (original) The method of claim 13, wherein said step of enabling includes logic to enable a required number of slices to execute the operation.

17. (currently amended) A processor comprising:

a plurality of slices, each of which is a portion of a full width word of the processor, wherein each slice comprises a portion of a register file, a portion of functional units, a portion of a memory path, a portion of a cache memory, and a portion of other resources required to perform operations in the processor,

logic to save and retrieve a width and selected bits of operands used to perform an operation in the processor,

logic to determine a prediction of arithmetic overflow when performing the operation, based upon the width and the selected bits of the operands used to perform the operation,

logic to determine a number of slices required to perform the operation based upon the width of one or more operands, the functionality of the operation, and the prediction of arithmetic overflow,

logic to activate the determined number of slices required to perform the operation.

18. (original) The processor of claim 17, including logic to determine the width of the result of the operation,

circuitry to store said width and selected bits of said result, and

circuitry to store said result.